Eddi: Interactive Topic-based Browsing of Social Status Streams

Michael S. Bernstein[1], *Bongwon Suh*[2], *Lichan Hong*[2], *Jilin Chen*[3], *Sanjay Kairam*[2], *Ed H. Chi*[2]

[1] MIT CSAIL
32 Vassar St., Cambridge MA
msbernst@mit.edu

[2] Palo Alto Research Center
3333 Coyote Hill, Palo Alto CA
{suh, hong, kairam, echi}@parc.com

[3] University of Minnesota
200 Union St., Minneapolis MN
jilin@cs.umn.edu

ABSTRACT

Twitter streams are on overload: active users receive hundreds of items per day, and existing interfaces force us to march through a chronologically-ordered morass to find tweets of interest. We present an approach to organizing a user's own feed into coherently clustered trending topics for more directed exploration. Our Twitter client, called Eddi, groups tweets in a user's feed into topics mentioned explicitly or implicitly, which users can then browse for items of interest. To implement this topic clustering, we have developed a novel algorithm for discovering topics in short status updates powered by linguistic syntactic transformation and callouts to a search engine. An algorithm evaluation reveals that search engine callouts outperform other approaches when they employ simple syntactic transformation and backoff strategies. Active Twitter users evaluated Eddi and found it to be a more efficient and enjoyable way to browse an overwhelming status update feed than the standard chronological interface.

ACM Classification: H5.2. Information interfaces and presentation (e.g., HCI): User interfaces.

General Terms: Design, Human Factors

Author Keywords: Twitter, topic clustering, social streams

INTRODUCTION

Social status streams threaten to become a torrent. As our information consumption grows to include microblogging services like Twitter and Facebook, tools struggle to keep up. Active Twitter users can easily receive over 1000 tweets in their streams each day, covering a wide variety of topics like research, design, dinner, Apple, Twitter, and the latest Internet meme. Twitter users often express a desire to filter the torrent down to just a single current, focusing on just tweets on a single topic or muting a popular but overplayed topic. Such interfaces do not exist, in large part because 140 characters is too short for text processing algorithms to index. Users must instead trim their follow lists or grimly resign themselves to missing interesting posts.

UIST'10, October 3–6, 2010, New York, New York, USA.

In this work we introduce *an interactive topic browser for Twitter feeds and an algorithm that makes Twitter topic browsing feasible*. When a feed is overwhelmed with large numbers of tweets, topic browsing can help the user gain a foothold—especially when the feed is dominated by informational tweets [19]. Topic-based browsing approaches have seen success in information-oriented tasks within search [6, 17], document exploration [11, 16] and blog interfaces [1], but have yet to be applied to social streams due to difficulty in modeling short pieces of content. Our topic-oriented Twitter browser, Eddi, groups the user's feed into coherent threads of conversation such as research, design, Microsoft, and Kanye West (Figure 1).

Eddi's approach enables users to manage their feed in ways that have been difficult so far: rather than missing the hundreds or thousands of tweets arriving between log-ins, Eddi users see an overview of the stream since last time and decide what to explore. The system is a trending topics interface for your own social network, threading related tweets together into coherent conversations.

To enable this interface, we have developed *a novel technique called TweeTopic that uses search engines as a distributed knowledge base*. Using TweeTopic, the tweet "macbook died, but the Genius guys gave me a new 1!" will associate not just with topics explicitly mentioned like *MacBook*, but also ones obliquely referenced like *Genius Bar* or implied like *Apple*. Existing techniques (e.g. term frequency – inverse document frequency (TF-IDF) [23], topic modeling (LDA) [3, 21]) struggle with tweets as 140 characters is often too little to give the algorithm reliable signals about important words. Our work observes that tweets are approximately similar to search queries. It transforms tweets via part-of-speech analysis [2] and inverse document frequency (IDF) metrics into versions more amenable to search engines; it then uses the returned pages to bootstrap additional knowledge about the tweet. TweeTopic can complete this calculation within seconds of receiving the tweet and does not need topics defined beforehand.

Eddi and TweeTopic are influenced by past work on information management in blogs [1], topic browsing (e.g., [17]), and topic identification algorithms (e.g., [3, 8, 23]). We extend this research into the microblogging domain, where information overload is so overwhelming that most users do not even try to read everything.

Figure 1. Eddi is a topic-oriented browsing interface for Twitter. Clockwise from upper right is the tag cloud, timeline (hidden in another tab), the topic dashboard, and the navigational list.

The remainder of this paper is organized as follows: We first present related work in information stream browsing interfaces. We then introduce our system, Eddi, and describe the algorithm used to generate topics. We detail a comparison of TweeTopic to other topic detection approaches and a usability evaluation of the system.

RELATED WORK

Eddi and TweeTopic build on related work in three areas: microblogging, topic browsing interfaces, and topic modeling algorithms.

Twitter

The microblogging service Twitter has exploded in popularity since its 2006 launch. Users can choose to follow anyone else and receive every status update ("tweet") shared by that person. Java et al. found major trends on the service to be daily chatter, conversations, information sharing and news reporting [15]. Naaman et al. manually coded tweets to quantify this breakdown, revealing that information sharing (22%), opinions (~25%), random thoughts (~25%), and personal status (~40%) make up the vast majority of tweets [19]. In the Discussion section, we will consider how our work is impacted by these categories. Honeycutt [13] investigated the conversational aspects of Twitter, which might help us segment topics.

Though network-scale analyses of Twitter are increasingly common (e.g., [15]), the literature on microblog consumption is still nascent. Zhang et al. [27] studied the use of Yammer, an enterprise-oriented Twitter service, inside a Fortune 500 company. The authors found a strong correlation between a user's ability to find relevant information on Yammer and the service's perceived usefulness. Ehrlich and Shami [7] found microblogs increasingly being used as a real-time information source, but incoming volume became overwhelming. Both papers conclude that solving the noise and relevancy problems, as Eddi attempts to do, is an important challenge.

Topic Browsing

Eddi chooses a clustering approach due to the dynamic, broad nature of Twitter discussion. We introduce 'topic' as a facet by which users can browse their feed. Faceted browsing interfaces have outperformed hierarchical interfaces on a variety of search and exploration tasks [26]. However, topic-clustering systems are difficult to design. Hearst notes that while they are automatable and are more responsive to unknown inputs, topic-clustering interfaces are less understandable than compiled categories [11, 12].

Nonetheless, researchers have found advantages for topic-clustering interfaces in a variety of user scenarios. Dumais et al. found that classifying web pages into categories reduced time required in search tasks [6]. Indeed, in a longitudinal evaluation of a system called Findex, Käki found that clustered search categories were useful when relevant documents were far down in the results list [17].

Topic browsing is also beginning to make inroads into social computing interfaces such as blogs. Baumer and Fisher performed a topic analysis of blogger's blogrolls, focusing, as we do, on information in the user's attentional sphere rather than the entire information space [1]. MrTaggy offered a topic-based browsing interface over social bookmark data [16]. Other approaches such as visual analytics of tweet contents via word frequency visualizations (e.g., [20]) fall under this umbrella as well.

We draw on research visualizing and navigating large document collections (for a review, see Hearst [12]). We find it appropriate to visualize tweets using a tag cloud, since tag clouds are shown to be effective signalers of social activity [10]. The inclusion of analytics views such as the ThemeRiver [9] are also intended to encourage social data analysis, as advocated by Wattenberg and Kriss [25].

Topic Identification

Data mining researchers have begun extracting structure from social streams. Ramage, Liebling and Dumais recently trained LDA models from Twitter data [21], finding that they can use the generated vectors to recommend tweets. Leskovec et al. tracked textual memes online through the 2008 Presidential election [18]. Shamma et al. mapped tweet keywords onto the Presidential debate timeline using a TF-IDF-style metric [24].

Information retrieval research complements these data mining approaches. Noun phrases are known to be good topic markers in paragraph-long text descriptions in information retrieval (IR) research [2, 14]. However, there are too many noun phrases to show to a user, so Bendersky et al. perform machine learning on noun phrase rankings to improve accuracy [2]. Rather than utilize this heavyweight approach, we will take advantage of search engines' ability to rewrite queries to solve the ranking problem.

TweeTopic builds on research that uses external knowledge sources to label text. Dakka and Ipeirotis mined WordNet and Wikipedia to create facet hierarchies for news articles [5], and Gabrilovich and Markovich represented news articles as a weighted vector of Wikipedia-based concepts to compute similarity metrics [8]. Because many Twitter topics are breaking news or are not considered important enough to be in curated knowledge repositories, our work contributes the notion of turning to a search engine as a knowledge base. Tweets are also several orders of magnitude shorter than news articles, so we introduce text transformations to make topic identification more reliable. Sahami and Heilman used Google to compute similarity of two short text queries [22], and TweeTopic takes inspiration from this search engine-based approach.

DESIGN HYPOTHESES

We began with informal survey research of social status stream users on Mechanical Turk (N=78) and in-person interviews. Participants consistently cited topic as an important factor in whether a tweet was important to see. Users wanted to track important but uncommon topics (e.g., *hci research*), and filter out common but undesired topics (e.g., *lunch*). In particular, while weak-tie friends tweet about issues of interest, they also tweet about other topics and personal life issues that users would often rather filter.

We developed design principles from our observations:

- First, because tweets are tiny, they are as fast for users to preview as to read completely. They have no intermediate representation like a headline that can be used for search and news topic browsing interfaces. Thus, Eddi provides a quick brushing (mouseover) interaction and a topic dashboard, making it quick for users to skim tweets directly.
- Second, unlike news articles, tweets often blend commentary with information content. Eddi supports discussion by bringing commentary on a single topic together even if the tweets are distributed in time.
- Third, tweets are often intended to be consumed in real-time and have important temporal characteristics. We thus included a timeline that allows users to check for updates in the minutes, days, or weeks since they last checked Twitter. We also ensured that our algorithm can process a tweet within seconds.

EDDI: TOPIC-BASED BROWSING INTERFACE

Eddi is a novel interface for browsing Twitter streams that clusters tweets by topics trending within the user's own feed (Figure 1). Eddi is named after the phenomenon of river eddies: swirling stationary points in quickly moving streams. The system observes a user's Twitter stream and constructs these stationary points of interest for the user to explore. Unlike existing trending topic interfaces like Tweetmeme and Twitter's trending topics, Eddi identifies topics within the user's own feed and about single tweets. Even if only a single user on Twitter tweets about UIST, Eddi will recognize it and bring it to the user's attention.

The Eddi landing page (Figure 1) displays an at-a-glance dashboard view of the topics flowing through the user's Twitter stream. This dashboard view includes a topic tag cloud, a navigational list of topic categories, a timeline, and a set of recommended tweets on interesting topics. By default, Eddi shows the last 24 hours of the user's Twitter feed. A calendar widget is available for doing more detailed exploration of arbitrary date ranges.

Tag Cloud Overview

The tag cloud describes the major topical themes in the user's Twitter stream (Figure 1). Some example topics that appear often in the authors' tag clouds include *research*, *design*, *search*, *iPhone*, *Twitter*, and *jQuery*, as well as topics tied to newsworthy events such as *healthcare, SXSW,* and *iPad*. Topic size is scaled proportionally to the number of tweets on the topic. These terms describe the tweets inside each topic category, though the individual tweets need not explicitly contain the topic words. Any #hashtags in the feed are automatically added to the interface.

Figure 2. The topic detail view for the *google* topic. At top, tweets believed to belong to the topic. Below, tweets that Eddi assigned a lower confidence score to the *google* topic.

Figure 3. The Topic Dashboard previews a subset of the topics inline, determined by what the user tends to tweet about.

The tag cloud includes a brushing interface for previewing the contents of a category. When the user brushes over a topic keyword in the tag cloud, a detail tooltip appears displaying the number of tweets and the five most recent tweets in the topic. This interface is intended to give users the ability to peek inside a topic before committing to explore it. For topics with five or fewer tweets, the entire contents can be read quickly by mousing over the topics title.

Topic Detail View

By clicking on a topic in the tag cloud, navigational list, topic dashboard, or timeline, the user can elect to view all the tweets in that topic (Figure 2). These tweets are displayed reverse-chronologically, as in most Twitter clients today, to preserve implicit conversation threads. Underneath is a second list of tweets that Eddi believes to be weakly associated with the category.

Navigational List

The left column contains a complete list of topics in the feed, sorted by popularity (Figure 1). The list also provides the same brushing tooltip as the tag cloud, so a user can mouse over a topic name to see a compressed list of tweets

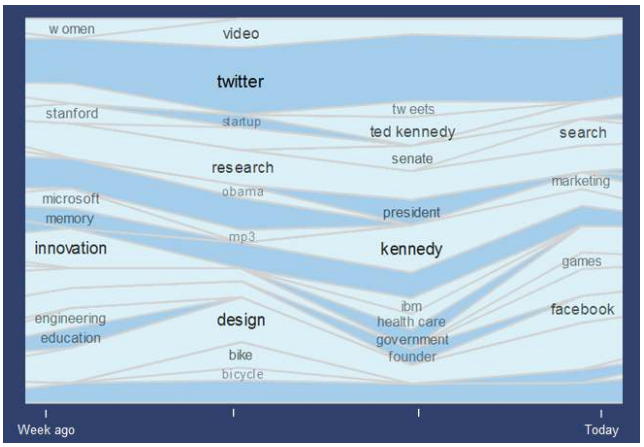


Figure 4. An example Eddi timeline showcases trending topics over the past week. Topics associated with Ted Kennedy trended up on the second day from the right, due to his death.

in the category. A search box at the top of the list allows for search over topic names in the list. In comparison to the tag cloud, which is limited to just the most popular topics, this list can be quite long to support serendipitous exploration of unpopular but interesting topics in the long tail.

Topic Dashboard

One drawback of an overview and detail interface is that the user must spend time absorbing the overview before seeing the tweets. We thus promote tweets on interesting topics to a summary dashboard on the homepage to make browsing more immediate (Figure 3). The dashboard shows up to five tweets chosen from the top ten topic categories. It contains a thumbnail of the author's profile picture, which we found suffices to identify many tweeters and allows recommended tweets to be displayed with few pixels.

To recommend personalized topics, we analyze the user's own tweets to find topics they talk about. We found that this heuristic works reasonably well for most users, similar to past research [4, 21]. When a topic in the user's interest profile appears in the feed, it is added to the dashboard regardless of the number of tweets in the topic. Thus, a topic of great interest such as *HCI* will be promoted to the top of the dashboard even if there is only a single tweet.

Timeline

The topic timeline, available in a separate tab next to the tag cloud, draws attention to temporal events in the feed (Figure 4). The timeline focuses on time-sensitive spikes by using a stacked graph visualization [9, 25]: the height of the strip at a particular time is based on the number of tweets in that topic. Topics that trend over time will grow in size at particular times and become more salient, such as news items, conferences, and announcements.

TWEETOPIC: TOPIC ASSIGNMENT ALGORITHM

In this section, we describe Eddi's topic assignment algorithm for short status updates. Such algorithms have many applications in real-time search and microblogging user interfaces. The TweeTopic algorithm labels a tweet

Apple


W00t! Snow Leopard gave me 10 gigs back!


RT @username: gmail is down, but the imap connection on my iphone still works (fingers crossed!)


My iPhone 3GS cracked-on-a-rock, @username's swam in a toilet, both repaired/replaced in 20 min @ Boylston Apple Store. Total cost: $0.

Obama


I think the most striking thing about Obama's speech + GOP response for casual listeners would be how much agreement there was.


RT @username: The fastest way to prove you are an idiot is to call the President a liar on live TV

Research


@username Congratulations on the CSCW best paper nomination!


Stanford scientists turn liposuction leftovers into embryonic-like stem cells: http://bit.ly/3GHsw9


CORRECTION: the deadline for submissions to the Graduate Student Consortium for TEI '09 is October 2 http://bit.ly/15D8Mv

Table I. An example set of categories and tweets in those categories, as classified by TweeTopic.

with a set of topic descriptors that can be used in the Eddi user interface. TweeTopic is easily implementable, does not require long training periods and is appropriate for both interactive applications and prototyping.

Current best practices for topic identification assume that indexed documents are of a decent length. This is an assumption that tweets fail almost by definition, being at most 140 characters long. Short length leads to poor topic analysis of tweets if we use traditional approaches. Specifically, traditional techniques tend to use word repetition as a measure of importance or term co-occurrence to identify related words. Word repetition approaches such as term frequency – inverse document frequency (TF-IDF) [23] suffer because users tend to remove word redundancy from a tweet to save space. Techniques based on co-occurrence matrices, such as Latent Dirichlet Allocation (LDA) [3, 21], are likewise debilitated by users compressing out similar words to gain space to insert their own opinions.

We can illustrate with an example tweet: "Ron Rivest cracks me up. It keeps me awake when algorithm design brings the LOLZ." In this tweet, no terms are repeated. Thus, TF-IDF is essentially just the IDF term. Since IDF looks for uncommon words, this approach produces bad topic words like *cracks*, *keeps*, *brings* and *LOLZ*.

TweeTopic Overview

Eddi's algorithm, TweeTopic, takes in a status update and outputs a list of ranked terms describing it. Thus, a tweet such as "Awesome article on some SIGGRAPH user interface work: http://bit.ly/30MJy" will output terms such as *animation, character, 3d, computer graphics*, *user interface* and *SIGGRAPH*. Table I shows example tweets along with their assigned topics.

The central intuition behind TweeTopic is that we can finesse a tweet to look like a search query, then use search engines to retrieve documents to expand our knowledge about the text. The algorithm proceeds in three steps: text transformation, search engine querying, and result mining.

Step 1: Text Transformation

TweeTopic's central approach is to utilize search engines as an external knowledge source; however, search engines expect short, direct queries as inputs. Tweets are more likely to be ungrammatical and rambling, so the first step in TweeTopic is to transform the update into keywords that might resemble a search query. We begin by heuristically transforming Twitter-specific markups to make the tweet appear more syntactical by removing terms such as RT and turning @username mentions into capitalized names.

We then adapt the notion that noun phrases are good topic markers in paragraph-long text descriptions [2, 14]. We use a maximum entropy part-of-speech tagger[1] to identify all the noun phrases in the tweet. In the tweet "Awesome article on some SIGGRAPH user interface work: http://bit.ly/30MJy", we identify the word "article" and the phrase "SIGGRAPH user interface work" for analysis.

Step 2: Query a Search Engine

The second step is to concatenate the noun phrases and send them as a query to a search engine. From the tweet in step 1, our query is "article SIGGRAPH user interface work." The tweet now resembles a reasonable search engine query. We can now use the search engine as a large knowledge base. The search engine will return a result set of documents that best captures the gist of the query.

Often, the query is over-constrained and the search engine returns fewer than ten or no results for our query. This occurs often when we identify many nouns in the tweet and thus have a long query. In these cases, we use *iterative backoff* to adjust the query until the search engine returns at least 10 results. We repeatedly remove from the query the word with the fewest occurrences on the web (these may be misspellings or mistakes that over-constrain the search) until we get a result. Then we start over, iteratively removing words with the most occurrences on the web (these may be generic noise terms). We finally average the result votes from the two runs. Our system uses the Yahoo! Build Your Own Search Service, or Y!BOSS[2], to issue queries.

Step 3: Identify Popular Terms in the Results

The final step in our algorithm identifies popular terms in the search results and assigns these as potential topics for the tweet. For each of the top ten returned results, we identify important terms in the webpage via a weighting scheme such as TF-IDF. Y!BOSS provides a list of about 20 TF-IDF-style key terms for each search result, which we use directly. For each page in the top ten results, we tally one

[1] http://nlp.stanford.edu/software
[2] http://developer.yahoo.com/search/boss

vote for each key term (topic) associated with the page. Topics with at least five votes are certified as valid descriptors of the tweet. Topics with fewer votes are retained as suggestions. We found through experimentation that five votes provided the right balance between accuracy and completeness. After this step, we are left with a set of terms describing the tweet and a 10-point score representing the number of search documents that voted for each term.

Combining Categories

Each tweet is now described with multiple topics, which for Eddi's purposes often gives us too many topics to display. We need the smallest number of topics that ensure all tweets are represented. This is a version of the set cover problem; we utilize a typical greedy solution that takes the largest topic grouping, assigns those tweets to that topic, and works its way to less popular topics while discarding topics whose tweets have been entirely covered by other topics. We allow tweets to exist in multiple categories.

EVALUATION

This paper introduces both an algorithm for topic detection and a topic-oriented user interface for social information streams such as Twitter feeds. In this section, we (1) benchmark TweeTopic against other topic detection approaches, and (2) compare Eddi to a typical chronological interface for consuming Twitter feeds.

Study 1: Algorithm Topic Detection Comparison

In this section we benchmark TweeTopic's performance to other algorithms, and to variations of itself. By doing so, we can learn which steps in the algorithm are most important and how they compare to the state of the art.

Algorithms

To follow, we describe the algorithms in our evaluation. We chose alternative algorithms that are used for topic detection and topic-based user interfaces.

Unigram Random: A simple baseline algorithm that removes stopwords from the tweet and then chooses a random word from the remaining text.

Inverse Document Frequency (IDF): The traditional approach to identifying important words is term frequency – inverse document frequency (TF-IDF), but tweets are so short that the TF term is very noisy. Thus, we compared to IDF on a web corpus. We included an algorithm that ranked words by high IDF (*Unigram High IDF*), preferring common terms on the web, and an algorithm that ranked words by low IDF (*Unigram Low IDF*), preferring uncommon terms on the web. Because IR research has found noun phrases to be good topic markers in freeform text [2, 14], we also included *Noun Phrase High IDF* and *Noun Phrase Low IDF*. In our results, all low IDF conditions outperformed their high IDF counterparts, so we will report only Low IDF.

Topic Modeling (LDA): Another approach to topic detection is Latent Dirichlet Allocation, or *LDA* [3], which has seen considerable success modeling topics in web articles and Twitter [21]. However, it generates distributions over words, so individual words and phrases recommended as topics may be less helpful in a user interface. Following Ramage et al. [21], we gathered 1.6 million tweets from a 24-hour period on March 18th-19th 2010 using Twitter's spritzer feed, which streams a pseudo-random 5% selection of all tweets via an API. We used a language detection toolkit[3] to identify 850,000 of those tweets as English. We then stemmed and removed stopwords from the tweets, removed terms occurring fewer than 30 times across the dataset, and removed the 40 most popular words from the dataset. We weighted words in each topic by P(topic | tweet) so that terms strongly associated with the highly probable topics were recommended.

We also prepared versions of TweeTopic that disabled the tweet transformation and the search iteration.

Transformed vs. Raw: One step in TweeTopic transforms the tweet to look like a search query. We call the variants that do syntax manipulation, *Transformed*. Other variants used the raw tweet text. TweeTopic also performs two distinct transformation stages: stopword removal, and noun phrase extraction. We want to separate the effect of these stages, so we distinguished *Y!BOSS Transformed Iterated* (only stopword removal) from *Y!BOSS Noun-Transformed Iterated* (stopword removal and noun phrase extraction).

Iterated vs. None: The search step in the algorithm uses an interated backoff strategy to adjust the query if the search fails. Those that use backoff we call *Iterated*; other versions issued only a single query. So, *Y!BOSS* sent the whole tweet to the search engine and did not use backoff, whereas *Y!BOSS Transformed Iterated* both performed syntax transformation and used the search backoff strategy.

Data

We gathered 100 random tweets that passed through the Twitter spritzer feed a few minutes after the tweets used to train the LDA algorithm. We recruited three human coders for topic rating, all heavy Twitter users. We again filtered out non-English tweets and tweets that the coders could not understand. Each algorithm generated its top five topic recommendations for each tweet. If five topics could not be generated, any missing topics were automatically scored as incorrect. As in typical IR relevancy judgment tasks, each coder judged every topic recommendation as relevant or not relevant. This coding resulted in 11 algorithms * 5 topics per algorithm = 55 coded topic recommendations per tweet and 500 ratings per algorithm. We calculated inter-rater agreement on this dataset using Fleiss's Kappa (κ=.70, z=85.06, p<0.001) which indicated substantial agreement.

Results

Because our outcomes are binary and not continuous, we used logistic regression analysis to predict the probability of a binary output. In our case, this binary output is "1" if the

[3] http://www.microsofttranslator.com/dev/

topic was rated as correct and "0" if not. We added control variables for the tweet, rater and topic rank for each algorithm (highest rank: 1, lowest rank: 5).

Results from Table II demonstrate that TweeTopic-style search approaches outperformed other algorithms when using transformation or iteration. The odds-ratio describes how many times more likely the tweet is to be correctly classified with that algorithm than by the control, *Unigram Random*. For example, using *Y!BOSS Noun-Transformed Iterated* means that the topic is 1.90 times as likely to be classified as correct than with *Unigram Random*. Thus, TweeTopic doubles the baseline performance.

These numbers are useful but do not tell us whether the algorithms differ significantly. Using Wald tests with Bonferroni correction ($p < 0.05$), we compared individual algorithms. *Y!BOSS Noun-Transformed Iterated* was the highest-performing, so we compared it to each other algorithm. *Y!BOSS Noun-Transformed Iterated* and *Y!BOSS Transformed Iterated* are statistically indistinguishable, but the algorithm statistically outperforms all others.

Search engine callouts were successful only if they used either iteration or transformation. With neither transformation nor iteration, the search often returned no results, leading to only one-tenth the accuracy as choosing a random non-stopword from the tweet. Combining both transformation and iteration outperformed either alone.

Counter to our hypothesis, while noun phrases are good markers for long queries [2] and paragraphs [14], they seem to be unnecessary for tweets. The simplest version of the algorithm with good performance only requires the removal of stopwords and Twitter lingo. Figure 5 tells a similar story when the algorithms are only asked for a single top-ranked prediction. Transformation and iteration lead to about 40% precision – about twice our baseline expectation and almost three times what an LDA topic model provides.

Latent Dirichlet Analysis generally did not provide meaningful topic terms. LDA identified words associated with words in the tweet, rather than meaningful topics. So, a tweet about Chinese vaccines in the news produced poor suggestions like *free*, *read,* and *via*. (It did find good topics as well, like *news*.) We believe that LDA will be much more useful in scenarios like personalization, where term vectors and large numbers of words are necessary [21].

Study 2: Browsing Your Own Twitter Feed

Eddi's design hypothesizes that a topic-oriented interface for Twitter can help users manage the overwhelming data in their feeds. To test this hypothesis, we conducted a laboratory study to get a broad sense of whether Eddi is subjectively better for browsing an overwhelming Twitter feed than standard reverse chronological list interfaces.

Eddi's challenge in this task is to offset the additional up-front cost of understanding the topic list and of navigating between topics. We gave users a short period of time to

Algorithm	Odds Ratio	z	p-value
Y!BOSS Noun-Transformed Iterated	1.90	6.78	< .001
Y!BOSS Transformed Iterated	1.88	6.70	< .001
Y!BOSS Transformed	1.43	3.77	< .001
Unigram Low IDF	1.31	2.80	< .01
Nouns Low IDF	1.28	2.61	< .01
Y!BOSS Iterated	1.23	2.19	< .05
Unigram Random (baseline)	(1.00)	n/a	n/a
LDA	0.41	-7.78	< .001
Y!BOSS	0.18	-12.08	<.001

Table II. Logistic regression coefficients. TweeTopic-style search techniques consistently outperformed others.

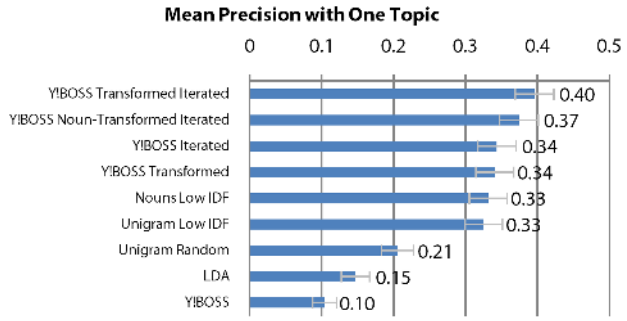


Figure 5. Transformation and iteration led to high precision.

browse their feed using both interfaces, and then asked them to compare the experiences. Our evaluation measured subjective feelings of *efficiency*, *effectiveness* and *satisfaction*, outlined by Hearst as important constructs for the evaluation of interfaces like Eddi [12]. Our hypothesis was that Eddi would be rated more effective and with higher satisfaction, without significantly decreasing efficiency, when compared to the chronological interface.

Participants

We recruited active Twitter users who checked the service daily, preferring those who followed over 100 other accounts. Fifteen users answered our call on Twitter in exchange for a small gratuity. Their ages ranged from 19 to 49 (median=29). Two were IT professionals, six were students, and the rest were scattered across careers like entertainment, content strategy, and finance. They reported checking Twitter several times a day, once an hour, or constantly. We found an average of 786 tweets per day in their feeds (σ=658), with a large right skew: the maximum was 2,840. The median participant followed 243 accounts (μ=382, σ=345). They used various Twitter clients, but Tweetie and the native web client were the most popular.

Interface Conditions

The study utilized a within-subjects design to directly compare Eddi to the standard reverse-chronological Twitter interface approach. To ensure that any observed differences were due to the Eddi interface and not to nuances of layout or coloring, users in the control condition saw a client that listed the tweets reverse-chronologically as in most Twitter clients today. Visually, this control interface looked very similar to the layout Eddi uses in the detail pane (Figure 2)

with all tweets in a vertical column. Constructing our own control interface allowed us to (1) display tweets from an arbitrary period of history, which most clients cannot do; (2) control for differences between participants' Twitter clients, and (3) remove the requirement that users click "More" to reveal more tweets. Participants noted that this interface felt extremely similar to the web interface.

In within-subjects studies, we do not want participants to see the same feed items in both conditions due to potential confounding learning effects. Thus, each trial featured a different 24-hour period from the user's Twitter timeline. We chose the date windows to be at least one week old to reduce memory effects, at least 4 days apart to reduce the likelihood that topics crossed over, and excluded weekends. Pilot studies suggested that this separation was sufficient to minimize undesired memory and learning effects, and that users remembered few of the tweets from this set.

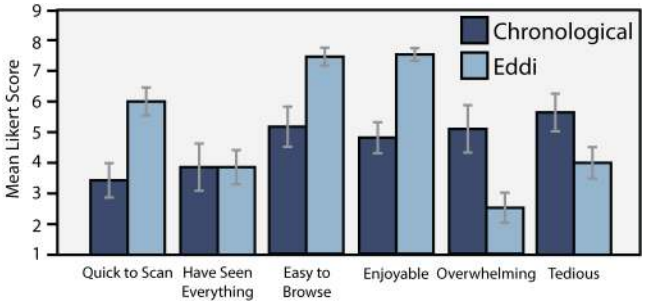
Procedure

The experimenter first trained participants on Eddi and the control interface. The participants' task was to browse their own Twitter feed, trying to absorb as much from each day's stream as possible. Participants were given three minutes in each trial in order to mimic a typical Twitter scenario: having a few moments of downtime to catch up on Twitter. To help us understand which tweets the user was reading, we used position on the screen as a proxy for attention, marking tweets as potentially viewed if they had remained in the browser window for at least two seconds. Tweets might appear onscreen in several ways: in the Topic Dashboard, in the mouse brushing previews, and in the topic detail pages. Tweets scrolled off-screen were not counted.

The study consisted of a total of six trials. In three of the trials, subjects used the reverse-chronological interface, and in the other three trials, subjects used Eddi. The interfaces alternated, and the order was counterbalanced between participants. We randomized the order of date windows.

After all trials were completed, participants reported subjective ratings on a 9-point Likert scale for each interface, with 1 meaning "strongly disagree" and 9 meaning "strongly agree." We used 9-point scales for consistency with previous work [26]. Our evaluation measured subjective outcomes outlined by Hearst as important constructs for the evaluation of interfaces like Eddi: *efficiency* (quick to scan), *effectiveness* (have seen everything), and *satisfaction* (easy to browse, enjoyable, overwhelming, and tedious) [12, 26].

To examine whether Eddi was helping users find interesting information, we asked participants to re-read and rate 300 tweets from previous trials as interesting or not interesting. We selected 150 tweets from one of the days previously browsed using Eddi, and 150 tweets from another day browsed using the chronological interface. We randomized the order of the 300 tweets and removed information about which interface the tweets had originally appeared in.



Metric	μ (Eddi)	μ (Chronological)	p-value
Quick to Scan	6.0	3.4	< .01
Have Seen Everything	3.9	3.9	.97
Easy to Browse	7.4	5.1	< .01
Enjoyable	7.5	4.8	< .01
Overwhelming	2.5	5.1	< .05
Tedious	4.0	5.6	< .05

Figure 6. Eddi fared better on all metrics except confidence in seeing everything desired in the feed.

Finally, the researcher led the participants through a semi-structured interview to draw out benefits and drawbacks.

Results: Subjective Assessment

To analyze the Likert scale comparisons, which are arguably non-continuous and non-normal, we utilized the nonparametric Wilcoxon Signed Ranks Test for paired samples. Eddi scored higher on the efficiency metric and all satisfaction metrics (Figure 6). It was quicker to scan, easier to browse, and more enjoyable. Effectiveness was approximately the same for both interfaces. These results supported our hypothesis that Eddi would improve the Twitter browsing experience for overwhelming feeds.

Efficiency was one of the greatest benefits of using Eddi. One participant explained: "Eddi helps me find things that I'm interested in, faster." Another participant offered, "It gives me a very quick way to have a first pass and to keep me from needing to read 140 characters about something I don't care about." The brushing interface for previewing categories provided a very fast mechanism for users to skim the contents of a category and decide to investigate.

Users found Eddi more enjoyable, easier to browse, less tedious and less overwhelming than the chronological feed. "This is kind of exactly what Twitter is missing," offered one participant who is following 1250 people. Another participant seconded, "I get bored faster with the traditional feed. There's way more stuff that I'm not interested in." A third user reacted by suggesting that an interface like Eddi would prompt him to combine his four Twitter accounts, which he had split due to overwhelming traffic.

Participants were more ambivalent about Eddi's effectiveness at coverage. When asked about the worst part of the Eddi interface, some participants who typically read every tweet in their feed mentioned the inability to know if they had seen "everything". The chronological interface was simply "less enjoyable but more comprehensive." Others,

who didn't feel compelled to see every tweet, saw the abstraction as a positive trait: "With the serial feed I feel like I need to see everything. Here, I can quickly get a gestalt."

TweeTopic fared well in qualitative feedback. In semi-structured interviews, participants reported that tweets were accurately classified. A typical response: "Yeah — I didn't see much that made me say, 'Why is this here?'" The terms selected for display were at the right level of abstraction: participants were not in favor of broader terms such as *technology* or *entertainment*, nor did they want to split existing terms into more specific ones.

We expected to see that participants would view more tweets in the chronological condition than in the Eddi condition, because it is faster to skim and scroll than to grasp and then navigate a set of categories. To test this, we performed repeated-measures ANOVA with number of tweets seen as the dependent variable and study condition as the independent variable, with each trial as an observation. The effect was significant ($F(1, 67) = 65.91$, $p<.001$), with the average participant viewing 80 tweets ($\sigma=37$) with Eddi and 145 tweets ($\sigma=75$) in the chronological condition. This suggests that, for our participants, Eddi's benefits subjectively compensated for viewing *half* as many tweets on average.

Results: Precision and Recall

By recording a sample of the tweets that participants found interesting and the tweets they viewed in each interface, we can be more quantitative about Eddi's effect on browsing.

We calculated *precision*, the percentage of all tweets viewed that the user found interesting in our rated sample, and *recall*, the percentage of all interesting tweets from our rated sample that the user saw while browsing. The mean precision across all participant trials was .19 ($\sigma=.15$), while the mean recall was .40 ($\sigma=.25$). We utilized repeated measure ANOVAs with precision and recall as the dependent variables and interface as independent variable.

Eddi users were more effective at finding tweets they were interested in seeing. Eddi significantly increased users' ability to read only tweets they want to see: precision was significantly higher in the Eddi condition ($\mu=.253$, $\sigma=.183$) than in the control condition ($\mu=.146$, $\sigma=.088$), ($F(1, 12) = 7.41$, $p < .05$, $\eta^2=.11$). In the Eddi condition, approximately one in four tweets seen were later rated as interesting; the chronological view was near one in seven. Recall was not significantly impacted ($F(1, 12) = 0.06$, n.s.), meaning that participants found the same number of interesting tweets in both conditions even though they viewed nearly twice as many in the chronological condition. Eddi thus allows users to consume a 'purer' Twitter stream without sacrificing coverage of interesting items.

Limitations

There are some clear limitations to our laboratory evaluation: (1) Users had access to our clients for a limited time, making it difficult to extrapolate conclusions on how the tool might be used longitudinally. For example, longitudinal use amongst the authors has found that topics tend to be stable over time, enabling the user to pin favorite topics to the dashboard. (2) Users were viewing the history of their feed rather than tweets they had never seen before, making our task slightly less realistic. However, because we selected Twitter users with active feeds, our participants had often not seen many tweets shown in the study.

DISCUSSION

Eddi was developed with an eye toward information-sharing tweets: tweets that break news, post URLs, or comment on events [19]. These tweets are most likely to share topics and contain stronger topic signals. However, topics can cross-cut tweet categories: in the topic *Apple*, tweets in all 3 categories of: news ("new iPhone announced!"), opinions ("I love my iPhone!"), and personal status ("my iPhone saved me – got me downtown in 15min!") are common. It is an open question how much a topic-oriented interface benefits users whose feeds consist mainly of personal status updates; however, our user study included all types of tweets, so we have some support that Eddi remains useful in these situations.

It may seem puzzling that our user study found that the topic organization was meaningful and accurate when the best algorithm had only 40% precision. How could these algorithms lead to a good user experience? Our experience is that Eddi's *Combining Categories* set cover algorithm that happens in Step 3 is mainly responsible. The algorithm strongly prefers topic categories that are shared between tweets. It is far more likely than 40% that a topic independently shared by two or more tweets will be correct. Thus, Eddi succeeds in part because it focuses on a user's entire network neighborhood at once. In future work we hope to explore this notion in more depth: what is the relationship between number of tweets and performance?

Eddi is best suited for feeds with more items than can be read. Users with small numbers of followees and users tracking groups closely may not gain as much benefit from Eddi. We believe that a mass of undifferentiated low-priority follows may be the right job for the system. The user can explore tweets from people who are interesting but do not need to be tracked closely, or followed only when they discuss certain topics. "Say I wanted a *tea* category," one participant explained. "I don't care about what some people say *not* about tea, but I definitely care about what they say about tea." We intend to pursue designs along these lines, such as a "Follow When…" system that allows users to follow accounts for particular topics. For example, "Follow @GuyKawasaki when he tweets about social computing; ignore the rest."

Search-based approaches currently still fail with some classes of tweets. For example, in non-literal tweets, quotes, or philosophizing, TweeTopic often acted literally; e.g., "I'm hungry as a hostage over here. Damn!" was tagged as *hostage*. The system could get distracted by strong Internet

search presences: the tweet “it’s raining” was tagged with *song* and *music* because of the famous song “It’s Raining Men”. Missing context could also throw off the algorithm: knowing that the tweet was from the day after St. Patrick’s day would have helped decipher “I wonder how many people went to jail last nite drunk wakin up pissed and hungova #rudeawakening”. Another route to improvement would be to expand links and hashtags in the tweets.

All clustering interfaces face the challenge that a single large cluster might dominate the display. For example, a user following only jQuery developers may have a rather monolithic feed. However, there are standard steps to avoid this problem: agglomerative clustering, for example, is more robust to the large-cluster problem.

CONCLUSION

We have described a novel interface called Eddi that allows users to explore overwhelming status update streams according to topics of interest. We introduced a simple, novel topic detection algorithm that uses noun-phrase detection and a search engine as an external knowledge base. The algorithm, TweeTopic, outperforms comparable topic detection algorithms on a rating task. In a user study, we have found that Eddi is more enjoyable and more efficient to browse than the traditional chronological Twitter interface. Subjects view a significantly lower percentage of undesired, irrelevant tweets, without sacrificing the total number of interesting tweets they see. We view Eddi as a promising first step toward building an intuitive, effective set of tools for absorbing and manipulating social status streams. These streams already include Facebook, Twitter, LinkedIn, and FriendFeed, and are sure to only grow in number.

ACKNOWLEDGMENTS

We thank PARC’s ASC group, David Karger, Rob Miller, our user study participants, and the anonymous reviewers.

REFERENCES

1. Baumer, E. and Fisher, D. Smarter Blogroll: An Exploration of Social Topic Extraction for Manageable Blogrolls. *HICSS '08*, IEEE Press (2008).
2. Bendersky, M. and Croft, W.B. Discovering key concepts in verbose queries. *SIGIR '08*, ACM Press (2008).
3. Blei, D.M., Ng, A.Y., and Jordan, M.I. Latent Dirichlet Allocation. *Journal of Machine Learning Research 3*, 4-5 (2003), 993-1022.
4. Chen, J., Nairn, R., Nelson, L., et al. Short and Tweet: Experiments on Recommending Content from Information Streams. *Proc. CHI '10*, ACM Press (2010).
5. Dakka, W. and Ipeirotis, P.G. Automatic Extraction of Useful Facet Hierarchies from Text Databases. *IEEE Data Engineering '08*, IEEE (2008).
6. Dumais, S., Cutrell, E., and Chen, H. Optimizing search by showing results in context. *CHI '01*, ACM Press (2001).
7. Ehrlich, K. and Shami, N. Microblogging Inside and Outside the Workplace. *ICWSM '10*, AAAI Press (2010).
8. Gabrilovich, E. and Markovitch, S. Computing semantic relatedness using wikipedia-based explicit semantic analysis. *IJCAI '07*, (2007), 6–12.
9. Havre, S., Hetzler, E., Whitney, P., and Nowell, L. ThemeRiver: visualizing thematic changes in large document collections. *IEEE Trans. Vis. and Comp. Graphics 8*, 1 (2002), 9-20.
10. Hearst, M.A. and Rosner, D. Tag Clouds: Data Analysis Tool or Social Signaller? *HICSS '08*, (2008), 160-160.
11. Hearst, M.A. Clustering versus faceted categories for information exploration. *CACM 49*, 4 (2006), 59.
12. Hearst, M.A. *Search User Interfaces*. Cambridge University Press, 2009.
13. Honeycutt, C. and Herring, S.C. Beyond Microblogging: Conversation and Collaboration via Twitter. *HICSS '09*, IEEE (2009).
14. Hulth, A. Improved automatic keyword extraction given more linguistic knowledge. *EMNLP '03*, ACL (2003), 216-223.
15. Java, A., Song, X., Finin, T., and Tseng, B. Why We Twitter: Understanding Microblogging Usage and Communities. *WebKDD '07*, ACM Press (2007), 56-65.
16. Kammerer, Y., Nairn, R., Pirolli, P., and Chi, E.H. Signpost from the masses: learning effects in an exploratory social tag search browser. *CHI '09*, ACM Press (2009), 625–634.
17. Käki, M. Findex: search result categories help users when document ranking fails. *CHI '05*, ACM Press (2005), 131-140.
18. Leskovec, J., Backstrom, L., and Kleinberg, J. Meme-tracking and the dynamics of the news cycle. *KDD '09*, ACM Press (2009), 497-506.
19. Naaman, M., Boase, J., and Lai, C. Is it Really About Me? Message Content in Social Awareness Streams. *CSCW '10*, ACM Press (2010).
20. Paley, W.B. TextArc: Showing Word Frequency and Distribution in Text. *Ext. Proc. IEEE InfoViz*, IEEE (2002).
21. Ramage, D., Dumais, S., and Liebling, D. Characterizing Microblogs with Topic Models. *ICWSM '10*, AAAI Press (2010).
22. Sahami, M. and Heilman, T.D. A web-based kernel function for measuring the similarity of short text snippets. *WWW '06*, ACM Press (2006), 377-386.
23. Salton, G. and Buckley, C. Term-weighting approaches in automatic text retrieval. *Information Processing and Management 24*, 5 (1988), 513-523.
24. Shamma, D.A., Kennedy, L., and Churchill, E.F. Tweet the debates: understanding community annotation of uncollected sources. *Multimedia '09*, (2009).
25. Wattenberg, M. and Kriss, J. Designing for social data analysis. *IEEE Trans. Viz. and Comp. Graphics 12*, 4 (2006), 549-57.
26. Yee, K., Swearingen, K., Li, K., and Hearst, M. Faceted metadata for image search and browsing. *CHI '03*, ACM Press (2003), 401-408.
27. Zhang, J., Qu, Y., Cody, J., and Wu, Y. A case study of micro-blogging in the enterprise: use, value, and related issues. *Proc.*, ACM Press (2010), 243-252.